V Binding Text

<template>

  <div id="app">

    <h1>{{ great }} {{ name }}</h1>

    <div *v-text*="channel"></div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      great: "Hello",

      name: "Diky Prayoga",

      channel: "Codevolution",

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

V BINDING HTML

<template>

  <div id="app">

    <h1>{{ great }} {{ name }}</h1>

    <div *v-html*="channel"></div>

    <div *v-html*="hack"></div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      great: "Hello",

      name: "Diky Prayoga",

      channel: "<b>Codevolution<b/>",

      hack: `<a href="#" onclick="alert('You have been hacked!')">Win a Prize!</a>`,

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

V BINDING CLASS

<template>

  <div id="app">

    <h1 *v-bind*:*class*="underline">This Is Underlined</h1>

    <h2 *class*="underlined" *v-bind*:*class*="status">Status</h2>

    <h2 *v-bind*:*class*="isPromoted && 'promoted'">Promoted Movie</h2>

    <h2 *v-bind*:*class*="isSoldOut ? 'sold-out' : 'new'">Soldout? Movie</h2>

    <h2 *v-bind*:*class*="['new', 'promoted']">Newly promoted movie</h2>

    <h2

*v-bind*:*class*="[isPromoted && 'promoted', isSoldOut ? 'sold-out' : 'new']"

    >

      Array Conditional Movie

    </h2>

    <h2

*v-bind*:*class*="{

        promoted: isPromoted,

        new: !isSoldOut,

        'sold-out': isSoldOut,

      }"

    >

      Object Conditional Movie

    </h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      isDisabled: true,

      underline: "underlined",

      status: "success",

      isPromoted: true,

      isSoldOut: true,

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

.underlined {

  text-decoration: underline;

}

.promoted {

  font-style: italic;

}

.sold-out {

  color: red;

}

.new {

  color: green;

}

</style>

V BINDING STYLE & V SHORTHAND BINDING

<template>

  <div id="app">

    <h1 :*class*="underline">This Is Underlined</h1>

    <h2 *class*="underlined" :*class*="status">Status</h2>

    <h2 :*class*="isPromoted && 'promoted'">Promoted Movie</h2>

    <h2 :*class*="isSoldOut ? 'sold-out' : 'new'">Soldout? Movie</h2>

    <h2 :*class*="['new', 'promoted']">Newly promoted movie</h2>

    <h2 :*class*="[isPromoted && 'promoted', isSoldOut ? 'sold-out' : 'new']">

      Array Conditional Movie

    </h2>

    <h2

      :*class*="{

        promoted: isPromoted,

        new: !isSoldOut,

        'sold-out': isSoldOut,

      }"

    >

      Object Conditional Movie

    </h2>

    <h2

      :*style*="{

        color: highlightColor,

        'font-size': headerSize + 'px',

        padding: '20px',

      }"

    >

      Inline Style

    </h2>

    <h2 :*style*="headerStyleObject">Style Object</h2>

    <div :*style*="[baseStyleObject, successStyleObject]">Success Object</div>

    <div :*style*="[baseStyleObject, dangerStyleObject]">Danger Object</div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      isDisabled: true,

      underline: "underlined",

      status: "success",

      isPromoted: true,

      isSoldOut: true,

      highlightColor: "orange",

      headerSize: 50,

      headerStyleObject: {

        color: "orange",

        fontSize: "50px",

        padding: "20px",

      },

      baseStyleObject: {

        fontSize: "50px",

        padding: "10px",

      },

      successStyleObject: {

        color: "green",

        backgroundColor: "lightgreen",

        border: "1px solid green",

      },

      dangerStyleObject: {

        color: "white",

        backgroundColor: "red",

        border: "1px solid red",

      },

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

.underlined {

  text-decoration: underline;

}

.promoted {

  font-style: italic;

}

.sold-out {

  color: red;

}

.new {

  color: green;

}

</style>

CONDITIONAL RENDERING (v-if, v-else-if, v-else)

<template>

  <div id="app">

    <h2 *v-if*="num === 0">The number is zero</h2>

    <h2 *v-else-if*="num < 0">The number is negative</h2>

    <h2 *v-else-if*="num > 0">The number is positive</h2>

    <h2 *v-else*>Not a number</h2>

    <template *v-if*="display">

      <h2>Diky</h2>

      <h2>Codevolution</h2>

      <h2>Vue</h2>

    </template>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      num: "HI",

      display: true,

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

CONDITIONAL RENDERING (v-show)

<template>

  <div id="app">

    <h2 *v-if*="num === 0">The number is zero</h2>

    <h2 *v-else-if*="num < 0">The number is negative</h2>

    <h2 *v-else-if*="num > 0">The number is positive</h2>

    <h2 *v-else*>Not a number</h2>

    <template *v-if*="display">

      <h2>Diky</h2>

      <h2>Codevolution</h2>

      <h2>Vue</h2>

    </template>

    <h2 *v-show*="showElement">USING V SHOW</h2>

    <h2 *v-if*="showElement">USING V IF</h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      num: "HI",

      display: true,

      showElement: false,

    };

  },

  components: {},

};

</script>

<style>

</style>

LIST RENDERING (v-for)

<template>

  <div id="app">

    <h2 *v-for*="(name, index) in names" :*key*="index">{{ index }} {{ name }}</h2>

    <h2 *v-for*="name in fullNames" :*key*="name.first">

      {{ name.first }} {{ name.last }}

    </h2>

    <div *v-for*="actor in actors" :*key*="actor.name">

      <h2>{{ actor.name }}</h2>

      <h3 *v-for*="movie in actor.movies" :*key*="movie">{{ movie }}</h3>

    </div>

    <h2 *v-for*="(info, key, index) in myInfo" :*key*="info">

      {{ index }} {{ key }} {{ info }}

    </h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      names: ["Diky", "Difa", "Hendra"],

      fullNames: [

        { first: "Diky", last: "Prayoga" },

        { first: "Difa", last: "Nugraha" },

        { first: "Rahmat", last: "Darmawan" },

      ],

      actors: [

        {

          name: "Diky Prayoga",

          movies: ["Batman", "The Prestige"],

        },

        {

          name: "Di Caprio",

          movies: ["Titanic", "Inception"],

        },

      ],

      myInfo: {

        name: "Diky Prayoga",

        student: "STMIK AMIKBANDUNG",

      },

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

LIST AND KEYS

<template>

<template v-for="name in names" :key="name">

<h2>{{ name }}</h2>

<input placeholder="Last name" />

<hr />

</template>

<button @click="shuffle">Shuffle!</button>

</template>

<script>

import \_ from "lodash";

export default {

name: "App",

data() {

return {

names: ["Bruce", "Clark", "Diana", "Barry"],

};

},

methods: {

shuffle() {

console.log(this.names);

this.names = \_.shuffle(this.names);

},

},

};

</script>

<style>

</style>

CONDITIONAL LIST RENDERING

<template>

  <div id="app">

    <div *v-for*="name in names" :*key*="name">

      <h2 *v-if*="name === 'Diky'">{{ name }}</h2>

    </div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      names: ["Diky", "Difa", "Hendra"],

    };

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

METHODS

<template>

  <div id="app">

    <h2>add methods - {{ add() }}</h2>

    <h2>addWithParameter methods - {{ addWithParameter(10, 10, 10) }}</h2>

    <h2>multiply methods - {{ multiply(10) }}</h2>

    <h2>multiply methods - {{ multiply(baseValue) }}</h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      baseMultiplier: 5,

      baseValue: 2,

    };

  },

  methods: {

    add() {

      return 2 + 3 + 5;

    },

    addWithParameter(a, b, c) {

      return a + b + c;

    },

    multiply(num) {

      return num \* this.baseMultiplier;

    },

  },

  components: {},

};

</script>

<style>

</style>

EVENT HANDLING

<template>

  <div id="app">

    <h1>{{ name }}</h1>

    <button *v-on*:*click*="name = 'Difa Nugraha'">Change Name</button>

    <h2>{{ count }}</h2>

    <div>

      <button *v-on*:*click*="increment(1)">Increment 1</button>

      <button *v-on*:*click*="increment(5)">Increment 5</button>

      <button *v-on*:*click*="decrement(1)">Decrement 1</button>

      <button *v-on*:*click*="decrement(5)">Decrement 5</button>

    </div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      name: "Diky Prayoga",

      count: 0,

    };

  },

  methods: {

    increment(num) {

      this.count += num;

    },

    decrement(num) {

      this.count -= num;

    },

  },

  components: {},

};

</script>

<style>

</style>

EVENT HANDLING PART 2

<template>

  <div id="app">

    <h1>{{ name }}</h1>

    <button *v-on*:*click*="changeName($event), increment(1, $event)">

      Change Name

    </button>

    <h2>{{ count }}</h2>

    <div>

      <button @*click*="increment(1, $event)">Increment 1</button>

      <button @*click*="increment(5)">Increment 5</button>

      <button @*click*="decrement(1)">Decrement 1</button>

      <button @*click*="decrement(5)">Decrement 5</button>

    </div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      name: "Diky Prayoga",

      count: 0,

    };

  },

  methods: {

    changeName(event) {

      this.name = "Difa Nugraha";

      console.log("Event", event);

    },

    increment(num, event) {

      this.count += num;

      console.log("Event", event);

    },

    decrement(num) {

      this.count -= num;

    },

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

FORM HANDLING

<template>

  <div id="app">

    <div>

      <pre>

        {{ JSON.stringify(formValues, null, 2) }}

      </pre>

    </div>

    <form *action*="">

      <div>

        <label *for*="name">Name</label>

        <input *type*="text" id="name" *v-model*="formValues.name" />

      </div>

      <div>

        <label *for*="profile">Profile</label>

        <textarea id="profile" *v-model*="formValues.profileSummery"></textarea>

      </div>

      <div>

        <label *for*="country">Country</label>

        <select id="country" *v-model*="formValues.country">

          <option *value*="">Select a Country</option>

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

      <div>

        <label *for*="job-location">Job Location</label>

        <select id="job-locaiton" *multiple* *v-model*="formValues.jobLocation">

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

    </form>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      formValues: {

        name: "",

        profileSummery: "",

        country: "",

        jobLocation: [],

      },

    };

  },

  methods: {},

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

*/\* text-align: center; \*/*

  color: #2c3e50;

  margin-top: 60px;

}

label {

  font-weight: bold;

  display: flex;

  margin-bottom: 5px;

}

input + label {

  font-weight: bold;

  display: inline-flex;

  margin-right: 20px;

}

input[type="text"],

textarea,

select {

  display: block;

  width: 400px;

  padding: 6px 12px;

  font-size: 14px;

  line-height: 1.4;

  color: #555;

  background-color: #fff;

  background-image: none;

}

</style>

FORM HANDLING PART 2

<template>

  <div id="app">

    <div>

      <pre>

        {{ JSON.stringify(formValues, null, 2) }}

      </pre>

    </div>

    <form @*submit*="submitForm">

      <div>

        <label *for*="name">Name</label>

        <input *type*="text" id="name" *v-model*="formValues.name" />

      </div>

      <div>

        <label *for*="profile">Profile</label>

        <textarea id="profile" *v-model*="formValues.profileSummery"></textarea>

      </div>

      <div>

        <label *for*="country">Country</label>

        <select id="country" *v-model*="formValues.country">

          <option *value*="">Select a Country</option>

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

      <div>

        <label *for*="job-location">Job Location</label>

        <select id="job-locaiton" *multiple* *v-model*="formValues.jobLocation">

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

      <div>

        <input

*type*="checkbox"

          id="remoteWork"

*v-model*="formValues.remoteWork"

*true-value*="yes"

*false-value*="no"

        />

        <label *for*="remoteWork">Open to remote Work?</label>

      </div>

      <div>

        <label>Skill Set</label>

        <input

*type*="checkbox"

          id="html"

*value*="html"

*v-model*="formValues.skillSet"

        />

        <label *for*="html">HTML</label>

        <input

*type*="checkbox"

          id="css"

*value*="css"

*v-model*="formValues.skillSet"

        />

        <label *for*="css">CSS</label>

        <input

*type*="checkbox"

          id="javascript"

*value*="javascript"

*v-model*="formValues.skillSet"

        />

        <label *for*="javascript">JAVASCRIPT</label>

      </div>

      <div>

        <label>Years of Experience</label>

        <input

*type*="radio"

          id="0-2"

*value*="0-2"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="0-2">0-2</label>

        <input

*type*="radio"

          id="3-5"

*value*="3-5"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="3-5">3-5</label>

        <input

*type*="radio"

          id="6-10"

*value*="6-10"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="6-10">6-10</label>

        <input

*type*="radio"

          id="10"

*value*="10+"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="10">10+</label>

      </div>

      <div>

        <button>Submit</button>

      </div>

    </form>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      formValues: {

        name: "",

        profileSummery: "",

        country: "",

        jobLocation: [],

        remoteWork: "no",

        skillSet: [],

        yearsOfExperience: "",

      },

    };

  },

  methods: {

    submitForm(event) {

      event.preventDefault();

      console.log("form values", this.formValues);

    },

  },

  components: {},

};

</script>

<style>

#app {

}

label {

  font-weight: bold;

  display: flex;

  margin-bottom: 5px;

}

input + label {

  font-weight: bold;

  display: inline-flex;

  margin-right: 20px;

}

input[type="text"],

textarea,

select {

  display: block;

  width: 400px;

  padding: 6px 12px;

  font-size: 14px;

  line-height: 1.4;

  color: #555;

  background-color: #fff;

  background-image: none;

}

</style>

MODIFIER

<template>

  <div id="app">

    <div>

      <pre>

        {{ JSON.stringify(formValues, null, 2) }}

      </pre>

    </div>

    <form @*submit*.*prevent*="submitForm">

      <div>

        <label *for*="name">Name</label>

        <input *type*="text" id="name" *v-model*.trim.*lazy*="formValues.name" />

      </div>

      <div>

        <label *for*="profile">Profile</label>

        <textarea id="profile" *v-model*="formValues.profileSummery"></textarea>

      </div>

      <div>

        <label *for*="country">Country</label>

        <select id="country" *v-model*="formValues.country">

          <option *value*="">Select a Country</option>

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

      <div>

        <label *for*="job-location">Job Location</label>

        <select id="job-locaiton" *multiple* *v-model*="formValues.jobLocation">

          <option *value*="indonesia">Indonesia</option>

          <option *value*="malaysia">Malaysia</option>

          <option *value*="singapura">Singapura</option>

        </select>

      </div>

      <div>

        <input

*type*="checkbox"

          id="remoteWork"

*v-model*="formValues.remoteWork"

*true-value*="yes"

*false-value*="no"

        />

        <label *for*="remoteWork">Open to remote Work?</label>

      </div>

      <div>

        <label>Skill Set</label>

        <input

*type*="checkbox"

          id="html"

*value*="html"

*v-model*="formValues.skillSet"

        />

        <label *for*="html">HTML</label>

        <input

*type*="checkbox"

          id="css"

*value*="css"

*v-model*="formValues.skillSet"

        />

        <label *for*="css">CSS</label>

        <input

*type*="checkbox"

          id="javascript"

*value*="javascript"

*v-model*="formValues.skillSet"

        />

        <label *for*="javascript">JAVASCRIPT</label>

      </div>

      <div>

        <label>Years of Experience</label>

        <input

*type*="radio"

          id="0-2"

*value*="0-2"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="0-2">0-2</label>

        <input

*type*="radio"

          id="3-5"

*value*="3-5"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="3-5">3-5</label>

        <input

*type*="radio"

          id="6-10"

*value*="6-10"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="6-10">6-10</label>

        <input

*type*="radio"

          id="10"

*value*="10+"

*v-model*="formValues.yearsOfExperience"

        />

        <label *for*="10">10+</label>

      </div>

      <div>

        <label *for*="age">Age</label>

        <input

          @*keyup*.*enter*="submitForm"

*type*="number"

          id="age"

*v-model*.*number*="formValues.age"

        />

      </div>

*<!-- <div>*

*<button>Submit</button>*

*</div> -->*

    </form>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      formValues: {

        name: "",

        profileSummery: "",

        country: "",

        jobLocation: [],

        remoteWork: "no",

        skillSet: [],

        yearsOfExperience: "",

        age: null,

      },

    };

  },

  methods: {

    submitForm() {

*// event.preventDefault();*

      console.log("form values", this.formValues);

    },

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

*/\* text-align: center; \*/*

  color: #2c3e50;

  margin-top: 60px;

}

label {

  font-weight: bold;

  display: flex;

  margin-bottom: 5px;

}

input + label {

  font-weight: bold;

  display: inline-flex;

  margin-right: 20px;

}

input[type="text"],

textarea,

select {

  display: block;

  width: 400px;

  padding: 6px 12px;

  font-size: 14px;

  line-height: 1.4;

  color: #555;

  background-color: #fff;

  background-image: none;

}

</style>

BONUS DIRECTIVES

<template>

  <div id="app">

    <h2 *v-once*>{{ name }}</h2>

    <button @*click*="name = 'Difa Nugraha'">Change Name</button>

    <h2 *v-pre*>{{ name }}</h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      name: "Diky Prayoga",

    };

  },

  methods: {},

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

COMPUTED PROPERTIES

<template>

  <div id="app">

    <h2>Full Name - {{ firstName }} {{ lastName }}</h2>

    <h2>Computed Fullname - {{ fullName }}</h2>

    <button @*click*="items.push({ id: 4, title: 'Keyboard', price: 400 })">

      Add Item

    </button>

    <h2>Computed Total - {{ total }}</h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      firstName: "Diky",

      lastName: "Prayoga",

      items: [

        {

          id: 1,

          title: "TV",

          price: 100,

        },

        {

          id: 2,

          title: "Phone",

          price: 200,

        },

        {

          id: 3,

          title: "Laptop",

          price: 300,

        },

      ],

    };

  },

  methods: {},

  computed: {

    fullName() {

      return `${this.firstName} ${this.lastName}`;

    },

    total() {

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

COMPUTED PROPERTIES VS METHODS

<template>

  <div id="app">

    <h2>Full Name - {{ firstName }} {{ lastName }}</h2>

    <h2>Computed Fullname - {{ fullName }}</h2>

    <button @*click*="items.push({ id: 4, title: 'Keyboard', price: 400 })">

      Add Item

    </button>

    <h2>Computed Total - {{ total }}</h2>

    <h2>Method Total - {{ getTotal() }}</h2>

    <input *type*="text" *v-model*="country" />

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      firstName: "Diky",

      lastName: "Prayoga",

      items: [

        {

          id: 1,

          title: "TV",

          price: 100,

        },

        {

          id: 2,

          title: "Phone",

          price: 200,

        },

        {

          id: 3,

          title: "Laptop",

          price: 300,

        },

      ],

      country: "",

    };

  },

  methods: {

    getTotal() {

      console.log("getTotal Method");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

  },

  computed: {

    fullName() {

      return `${this.firstName} ${this.lastName}`;

    },

    total() {

      console.log("total computed property");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

COMPUTED PROPERTIES AND V BINDING FOR

<template>

  <div id="app">

    <h2>Full Name - {{ firstName }} {{ lastName }}</h2>

    <h2>Computed Fullname - {{ fullName }}</h2>

    <button @*click*="items.push({ id: 4, title: 'Keyboard', price: 400 })">

      Add Item

    </button>

    <h2>Computed Total - {{ total }}</h2>

    <h2>Method Total - {{ getTotal() }}</h2>

    <input *type*="text" *v-model*="country" />

    <div *v-for*="item in items" :*key*="item.id">

      <h2 *v-if*="item.price > 100">{{ item.title }} {{ item.price }}</h2>

    </div>

    <h2 *v-for*="item in expensiveItems" :*key*="item.id">

      {{ item.title }} {{ item.price }}

    </h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      firstName: "Diky",

      lastName: "Prayoga",

      items: [

        {

          id: 1,

          title: "TV",

          price: 100,

        },

        {

          id: 2,

          title: "Phone",

          price: 200,

        },

        {

          id: 3,

          title: "Laptop",

          price: 300,

        },

      ],

      country: "",

    };

  },

  methods: {

    getTotal() {

      console.log("getTotal Method");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

  },

  computed: {

    fullName() {

      return `${this.firstName} ${this.lastName}`;

    },

    total() {

      console.log("total computed property");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

    expensiveItems() {

      return this.items.filter((item) => item.price > 100);

    },

  },

  components: {},

};

</script>

<style>

</style>

COMPUTED SETTER

<template>

  <div id="app">

    <h2>Full Name - {{ firstName }} {{ lastName }}</h2>

    <h2>Computed Fullname - {{ fullName }}</h2>

    <button @*click*="changeFullName">Change Fullname</button>

    <button @*click*="items.push({ id: 4, title: 'Keyboard', price: 400 })">

      Add Item

    </button>

    <h2>Computed Total - {{ total }}</h2>

    <h2>Method Total - {{ getTotal() }}</h2>

    <input *type*="text" *v-model*="country" />

    <div *v-for*="item in items" :*key*="item.id">

      <h2 *v-if*="item.price > 100">{{ item.title }} {{ item.price }}</h2>

    </div>

    <h2 *v-for*="item in expensiveItems" :*key*="item.id">

      {{ item.title }} {{ item.price }}

    </h2>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      firstName: "Diky",

      lastName: "Prayoga",

      items: [

        {

          id: 1,

          title: "TV",

          price: 100,

        },

        {

          id: 2,

          title: "Phone",

          price: 200,

        },

        {

          id: 3,

          title: "Laptop",

          price: 300,

        },

      ],

      country: "",

    };

  },

  methods: {

    getTotal() {

      console.log("getTotal Method");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

    changeFullName() {

      this.fullName = "Rahmat Darmawan";

    },

  },

  computed: {

    fullName: {

      get() {

        return `${this.firstName} ${this.lastName}`;

      },

      set(value) {

        const names = value.split(" ");

        this.firstName = names[0];

        this.lastName = names[1];

      },

    },

    total() {

      console.log("total computed property");

      return this.items.reduce(

        (total, curr) => (total = total + curr.price),

        0

      );

    },

    expensiveItems() {

      return this.items.filter((item) => item.price > 100);

    },

  },

  components: {},

};

</script>

<style>

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

}

</style>

WATCHERS

<template>

  <div id="app">

    <h2>Volume Tracker (0-20)</h2>

    <h3>Current Volume - {{ volume }}</h3>

    <div>

      <button @*click*="volume += 2">Increase</button>

      <button @*click*="volume -= 2">Decrease</button>

    </div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      volume: 0,

    };

  },

  methods: {},

  computed: {},

  watch: {

    volume(newValue, oldValue) {

      if (newValue > oldValue && newValue === 16) {

        alert(

          "Listening to a high volume for a long time may damage your hearing"

        );

      }

    },

  },

  components: {},

};

</script>

<style>

</style>

IMMEDIATE AND DEEP WATCHERS

<template>

  <div id="app">

    <h2>Volume Tracker (0-20)</h2>

    <h3>Current Volume - {{ volume }}</h3>

    <div>

      <button @*click*="volume += 2">Increase</button>

      <button @*click*="volume -= 2">Decrease</button>

    </div>

    <input *type*="text" *v-model*="movie" />

    <input *type*="text" *v-model*="movieInfo.title" />

    <input *type*="text" *v-model*="movieInfo.actor" />

    <div>

*<!-- <button @click="movieList = movieList.push('Wonder Woman')">*

*Add Movie*

*</button> -->*

      <button @*click*="movieList = movieList.concat(['Wonder Woman'])">

        Add Movie

      </button>

    </div>

  </div>

</template>

<script>

*export* default {

  name: "App",

  data() {

    return {

      volume: 0,

      movie: "Batman",

      movieInfo: {

        title: "",

        actor: "",

      },

      movieList: ["Batman", "Superman"],

    };

  },

  methods: {},

  computed: {},

  watch: {

    volume(newValue, oldValue) {

      if (newValue > oldValue && newValue === 16) {

        alert(

          "Listening to a high volume for a long time may damage your hearing"

        );

      }

    },

    movie: {

      handler(newValue) {

        console.log(`Calling API with movie name = ${newValue}`);

      },

      immediate: true,

    },

    movieInfo: {

      handler(newValue) {

        console.log(

          `Calling API with movie title = ${newValue.title} and actor = ${newValue.actor}`

        );

      },

      deep: true,

    },

    movieList: {

      handler(newValue) {

        console.log(`Update List ${newValue}`);

      },

*// deep: true,*

    },

  },

  components: {},

};

</script>

<style>

</style>